Package 'hyper2'

May 16, 2019

Type Package

Title The Hyperdirichlet Distribution, Mark 2

Version 1.0-7

Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>

Description
A suite of routines for the hyperdirichlet distribution; supersedes the 'hyperdirichlet' package.

License GPL (>= 2)

LazyData yes

Depends R (>= 2.10), methods, magrittr, cubature

Suggests knitr

VignetteBuilder knitr

Imports Rcpp (>= 0.12.5), partitions

LinkingTo Rcpp

URL https://github.com/RobinHankin/hyper2.git

BugReports https://github.com/RobinHankin/hyper2/issues

NeedsCompilation yes

Author Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Repository CRAN

Date/Publication 2019-05-16 07:30:03 UTC

R topics documented:

`hyper2-package` *The Hyperdirichlet Distribution, Mark 2*

Description

A suite of routines for the hyperdirichlet distribution; supersedes the 'hyperdirichlet' package.

Details

The DESCRIPTION file:

Package:	hyper2
Type:	Package
Title:	The Hyperdirichlet Distribution, Mark 2

Version:	1.0-7
Authors@R:	person(given=c("Robin", "K. S."), family="Hankin", role = c("aut","cre"), email="hankin.robin@gmail.co
Maintainer:	Robin K. S. Hankin <hankin.robin@gmail.com>
Description:	A suite of routines for the hyperdirichlet distribution; supersedes the 'hyperdirichlet' package.
License:	GPL (>= 2)
LazyData:	yes
Depends:	methods, magrittr, cubature
Suggests:	knitr
VignetteBuilder:	knitr
Imports:	Rcpp (>= 0.12.5), partitions
LinkingTo:	Rcpp
URL:	https://github.com/RobinHankin/hyper2.git
BugReports:	https://github.com/RobinHankin/hyper2/issues
Author:	Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Index of help topics:

mult_grid	Kronecker matrix functionality
rowing	Rowing dataset, sculling
rrank	Random ranks
rugby	Super Rugby 2016 data
saffy	Choice matrix functionality
skating	Figure skating at the 2002 Winter Olympics
table_tennis	Match outcomes from repeated table tennis matches
tennis	Match outcomes from repeated doubles tennis matches
tidy	Keep or discard players
volley	Team sports

A generalization of the Dirichlet distribution, using a more computationally efficient method than the **hyperdirichlet** package. The software is designed for the analysis of order statistics and team games.

Author(s)

NA

Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

References

- R. K. S. Hankin (2010). "A Generalization of the Dirichlet Distribution", *Journal of Statistical Software*, 33(11), 1-18, http://www.jstatsoft.org/v33/i11/
- R. K. S. Hankin 2017. "Partial Rank Data with the hyper2 Package: Likelihood Functions for Generalized Bradley-Terry Models". *The R Journal* 9:2, pages 429-439.

Examples

```
H  <- hyper2(list(1,2,3,1:2,2:3),1:5)

data(chess)
maxp(chess)
```

B *Normalizing constant for the hyperdirichlet distribution*

Description

Uses numerical techniques for calculating the normalizing constant for the hyperdirichlet distribution

Usage

```
B(H, disallowed=NULL, give=FALSE, ...)
probability(H, disallowed=NULL, ...)
mgf(H, powers, ...)
dhyper2(P,H,...)
dhyper2_e(e,H,include.Jacobian=TRUE)
mean_hyper2(H, normalize=TRUE, ...)
Jacobian(e)
e_to_p(e)
p_to_e(p)
```

Arguments

H	Object of class hyper2
powers	Vector of length dim(x) whose elements are the powers of the expectation; see details section
disallowed	Function specifying a subset of the simplex over which to integrate; default NULL means to integrate over the whole simplex. The integration proceeds over p with disallowed(p) evaluating to FALSE
e,p	Vectors; see details
P	A vector of probabilities with unit rowsums
include.Jacobian	Boolean, indicated whether or not to include the Jacobian transformation in the evaluation
give	Boolean, with default FALSE meaning to return the value of the integral and TRUE meaning to return the full output of adaptIntegrate()
normalize	Boolean, indicates whether return value of mean_hyper2() is normalized to have unit sum
...	Further arguments passed to adaptIntegrate()

Details

- Function B() returns the normalizing constant; dhyper2(p,H)/B(H) is a probability density function. Internally, p is converted to e (by e_to_p()) and the integral proceeds over a hypercube.
- Function probability() gives the probability of an observation from a hyperdirichlet distribution satisfying !disallowed(p).
- Function mgf() is the moment generating function, taking an argument that specifies the powers of p needed: the expectation of $\prod_{i=1}^{n} p_i^{\text{powers}[i]}$ is returned.
- Function mean_hyper2() returns the mean value of the hyperdirichlet distribution. This is computationally slow (consider maxp() for a measure of central tendency). The function takes a normalize argument, not passed to adaptIntegrate(): this is Boolean with FALSE meaning to return the value found by integration directly, and default TRUE meaning to normalize so the sum is exactly 1

Value

- Function `B()` returns a scalar: the normalization constant
- Function `mean()` returns a k-tuple with unit sum
- Function `mgf()` returns a scalar equal to the expectation of `p^power`
- Functions `is.proper()` and `validated()` return a Boolean
- Function `probability()` returns a scalar, a (Bayesian) probability

Note

The adapt package is no longer available on CRAN; from 1.4-3, the package uses `adaptIntegrate` of the cubature package.

Author(s)

Robin K. S. Hankin

See Also

`loglik`

Examples

```
data(chess)
mean_hyper2(chess)
maxp(chess)

# disallowed argument typically means slow run times; use high tol for speed
probability(chess,disallowed=function(p){p[1]>p[2]},tol=0.1)
probability(chess,disallowed=function(p){p[1]<p[2]},tol=0.1)
```

---

change_pnames *Change pnames of a hyper2 object*

Description

Change the pnames attribute of a hyper2 object

Usage

```
change_pnames(H, new_pnames)
all_pnames(L)
```

Arguments

H	Object of class hyper2
L	List of hyper2 objects
new_pnames	Character vector of the new pnames

Note

Function change_pnames() is not a straightforward function. The intent is to enable one to add hyper2 objects with non-identical pnames attributes in a sensible way.

The standard use for change_names() is to add new players to a hyper2 object: so the new names must be a superset of the old, although the order may differ.

Function all_pnames() returns the union of the pnames of the hyper2 objects sent to it

Author(s)

Robin K. S. Hankin

Examples

```
x1 <- hyper2(list(1,1:2,1:3),1:3,pnames=letters[1:3])
x2 <- x1
pnames(x2) <- letters[3:1]

x1
x2
# At this point, note that x1 and x2 are algebraically identical; but:
x1==x2

## Variables x1 and x2 are different!  This is by design. The pnames
## attribute is just syntatic sugar, mathematically meaningless.

## Also note that you cannot add x1 and x2 because to do so is bad weirdness:
## Not run:
x1 + x2

## End(Not run)

## Here is a sensible use-case:

x1 <- hyper2(list(1,1:2,1:3),1:3,pnames=letters[1:3])
x2 <- hyper2(list(1,2),10:11,pnames=letters[4:5])

pnames(x1) <- letters[1:5]
pnames(x2) <- letters[1:5]
x1+x2


## NB:
```

```
pnames(x1) <- letters[5:1]
x1
```

`character_to_number` *Convert a character vector to a numeric vector*

Description

Convert string descriptions of competitors into their number

Usage

```
character_to_number(char, pnames)
char2num(char, pnames)
```

Arguments

`char`	Character vector to be converted
`pnames`	Names vector (usually `pnames(H)`)

Details

The internal mechanism of functions such as `ggol()`, and all the `C++` code, operates with the competitors labelled with a non-negative integer; it is then natural to refer to the competitors as `p1`, `p2`, etc.

However, sometimes the competitors have names (as in, for example, the `rowing` dataset). If so, it is more natural to refer to the competitors using their names rather than an arbitrary integer.

Function `character_to_number()` converts the names to numbers. If an element of `char` is not present in `pnames`, an error is returned.

Function `char2num()` is an easy-to-type synonym.

Author(s)

Robin K. S. Hankin

See Also

`order_likelihood`

Examples

```
H <- hyper2(pnames=letters[1:9])

# Use numbers:
H + order_likelihood(sample(9))


# Use letters:
H + order_likelihood(character_to_number(sample(letters[1:9]),pnames(H)))
```

---

chess *Chess playing dataset*

Description

A tally of wins and losses for games between three chess players: Topalov, Anand, Karpov

Usage

```
data(chess)
```

Details

This is a very simple dataset that can be used for illustration of hyper2 idiom.

The players are:

- Grandmaster Veselin Topalov. FIDE world champion 2005-2006; peak rating 2813
- Grandmaster Viswanathan Anand. FIDE world champion 2000-2002, 2008; peak rating 2799
- Grandmaster Anatoly Karpov. FIDE world champion 1993-1999; peak rating 2780

Observe that Topalov beats Anand, Anand beats Karpov, and Karpov beats Topalov (where "beats" means "wins more games than").

The games thus resemble a noisy version of "rock paper scissors".

The likelihood function does not record who played white; see karpov_kasparov_anand for such a dataset.

References

- http://www.chessbase.com/

See Also

karpov_kasparov_anand

Examples

```
data(chess)
maxp(chess)
```

`counterstrike` *Counterstrike*

Description

A kill-by-kill analysis of a counterstrike game.

Usage

```
data(counterstrike)
```

Details

E-sports are a form of competition using video games. E-sports are becoming increasingly popular, with high-profile tournaments attracting over 400 million viewers, and prize pools exceeding US$20m.

Counter Strike: Global Offensive (CS-GO) is a multiplayer first-person shooter game in which two teams of five compete in an immersive virtual reality combat environment.

CS-GO is distinguished by the ability to download detailed gamefiles in which every aspect of an entire match is recorded, it being possible to replay the match at will.

Statistical analysis of such gamefiles is extremely difficult, primarily due to complex gameplay features such as cooperative teamwork, within-team communication, and real-time strategic fluidity.

It is the task of the statistician to make robust inferences from such complex datasets, and here I analyze an influential match between "FaZe Clan" and "Cloud9", two of the most successful E-sports syndicates of all time, when they competed at Boston 2018.

Dataset `counterstrike` is a loglikelihood function for the strengths of ten counterstrike players. It can be created by executing the code in `inst/counterstrike.R`, which also shows how to create synthetic datasets for the purposes of comparison.

The probability model is identical to that of `NBA`: when a player kills (scores), this is taken to be a success of the whole team rather than the shooter.

The `counterstrike` object is created in `inst/counterstrike.R` (but is called "`H`" there, to avoid terrible confusion). There is quite a lot of extra documentation in that file.

Counterstrike dataset kindly supplied by Zachary Hankin.

References

- https://www.youtube.com/watch?v=XKWzlG4jDnI
- https://en.wikipedia.org/wiki/FaZe_Clan
- https://en.wikipedia.org/wiki/Cloud9

Examples

```
data(counterstrike)
dotchart(maxp(counterstrike))
```

`cplusplus` *Wrappers to c calls*

Description

Various low-level wrappers to c calls, courtesy of Rcpp

Usage

```
overwrite(L1, powers1, L2, powers2)
accessor(L,powers,Lwanted)
assigner(L,p,L2,value)
addL(L1,p1,L2,p2)
identityL(L,p)
evaluate(L,powers,probs)
differentiate(L, powers, probs, n)
```

Arguments

`L,L1,L2,Lwanted`	Lists with integer vector elements, used to specify the brackets of the hyperdirichlet distribution
`p,p1,p2,powers,powers1,powers2`	A numeric vector specifying the powers to which the brackets are raised
`value`	RHS in assignment, a numeric vector
`probs`	Vector of probabilities for evaluation of log-likelihood
`n`	Integer specifying component to differentiate with respect to

Details

These functions are not really intended for the end-user, as out-of-scope calls may cause crashes.

Value

These functions return a named List

Author(s)

Robin K. S. Hankin

`dirichlet` *Dirichlet distribution and generalizations*

Description

The Dirichlet distribution in likelihood (for p) form, including the generalized Dirichlet distribution due to Connor and Mosimann

Usage

```
dirichlet(powers, alpha, pnames=NA)
GD(alpha, beta, beta0=0, pnames=NA)
GD_wong(alpha, beta, pnames=NA)
```

Arguments

`powers`	In function `dirichlet()` a (named) vector of powers
`alpha,beta`	A vector of parameters for the Dirichlet or generalized Dirichlet distribution
`beta0`	In function `GD()`, an arbitrary parameter
`pnames`	Optional; a character vector specifying the names of p_1 through p_n

Details

These functions are really convenience functions.

Author(s)

Robin K. S. Hankin

See Also

`hyper2`

Examples

```
x1 <- dirichlet(c(a=1,b=2,c=3))
x2 <- dirichlet(c(c=3,d=4))


# will not work: x1+x2  ## probably not what you wanted
```

euro *Eurovision Song contest dataset*

Description

Voting patterns from Eurovision 2009

Usage

```
data(euro)
```

Format

A hyper2 object that gives a likelihood function.

Details

Object euro2009 is a hyper2 object that gives a likelihood function for the skills of the 18 competitor countries in semi-final 1 of the 2009 Eurovision Song contest. It is created by the self-contained code in inst/euro.R, which is heavily documented.

The reason for choosing this particular dataset is that Pat Altham (Statistical Laboratory, Cambridge) considered it with a view to discover similarities between voters. In the current analysis, the likelihood function euro2009 assumes their independence.

References

- Wikipedia contributors, "Eurovision Song Contest 2009—Wikipedia, The Free Encyclopedia", 2018, https://en.wikipedia.org/w/index.php?title=Eurovision_Song_Contest_2009&oldid=838723921 [Online; accessed 13-May-2018].
- P. M. E. Altham, personal communication

Examples

```
data(euro)

dotchart(maxp(euro2009))

x1 <- loglik(euro2009,indep(maxp  (euro2009)))
x2 <- loglik(euro2009,indep(equalp(euro2009)))

## if the competitors have equal strength, the asymptotic distribution
##   of 2*(x1-x2) is chi-square with 17 degrees of freedom:

pchisq(2*(x1-x2),df=size(euro2009)-1,lower.tail=FALSE)
```

Extract *Extract or replace parts of a hyper2 object*

Description

Extract or replace parts of a hyper2 object

Usage

```
## S3 method for class 'hyper2'
x[...]
## S3 replacement method for class 'hyper2'
x[index, ...] <- value
```

Arguments

x	An object of class hyper2
...	Further arguments, currently ignored
index	A list with integer vector elements corresponding to the brackets whose power is to be replaced
value	Numeric vector of powers

Details

These methods should work as expected, although the off-by-one issue might be a gotcha.

For the extract method, a numeric vector is returned but the elements may be in any order (the brackets and indeed the powers are not stored in any particular order).

The replace method, `H[L] <- value`, the index `L` is a list specifying the brackets to be overwritten; argument `value` is not recycled unless it is of length 1.

If the index argument is missing, viz `H1[] <- H2`, this is a special case. Argument `H1` must be a `hyper2` object, and the idiom effectively executes `H1[brackets(H2)] <- powers(H2)`, but more efficiently (note that this operation is well-defined even though the order of the brackets is arbitrary). This special case is included in the package because it has a very natural `C++` expression [function `overwrite()` in the `src/` directory] that was too neat to omit.

Altering (incrementing or decrementing) the power of a single bracket is possible using idiom like `H[x] <- H[x] + 1`; this is documented at `Ops.hyper2`, specifically `hyper2_sum_numeric()`.

Value

The extractor method returns a `hyper2` object, restricted to the elements specified

Note

Use `powers()` and `brackets()` to extract a numeric vector of powers or a list of integer vectors respectively.

Replacement idiom `H[x] <- val` cannot use non-trivial recycling. This is because the elements of `H` are stored in an arbitrary order, but the elements of `val` are stored in a particular order. Also see function `hyper2_sum_numeric()`.

Author(s)

Robin K. S. Hankin

See Also

`hyper2`,`Ops.hyper2`

Examples

```
data(chess)

chess[1]
chess[c(1,2)]
chess[list(1,1:2)]
chess[1] <- 5

a <- hyper2(pnames=letters[1:8])
a[sapply(seq_len(8),seq_len)] <- 1
a

b <- a
b[1] <- 1000
b[2] <- 2000
b[] <- a      # b[1] overwritten; b[2] preserved
```

`fillup` *Fillup function*

Description

Function `fillup()` concatenates a vector with a 'fillup' value to ensure a unit sum; if given a matrix, attaches a column so the rowsums are 1.

Function `indep()` is the inverse: it removes the final element of a vector, leaving only an independent set.

Usage

```
fillup(x)
indep(x)
```

Arguments

x	Numeric vector

Author(s)

Robin K. S. Hankin

See Also

equalp

Examples

```
fillup(c(1/2,1/3))

indep(c(1/2,1/3,1/6))
```

---

formula1 *Formula 1 dataset*

Description

Race results from 2017 Formula One World Championship

Usage

```
data(formula1)
```

Format

A hyper2 object that gives a likelihood function

Details

Object formula1 is a hyper2 object that gives a likelihood function for the strengths of the competitors of the 2017 Formula One World Championship. It is created from inst/formula1.txt by the code in inst/formula1.R, which is heavily documented.

References

"Wikipedia contributors", *2017 Formula One World Championship—Wikipedia, The Free Encyclopedia*, 2018. https://en.wikipedia.org/w/index.php?title=2017_Formula_One_World_Championship&oldid=839923210 [Online; accessed 14-May-2018]

Examples

```
data(formula1)
dotchart(maxp(formula1))
```

ggol *Order statistics*

Description

Various functions for calculating the likelihood function for order statistics

Usage

```
ggol(H, ...)
general_grouped_order_likelihood(H, ...)
choose_losers(H, all_players, losers)
choose_winners(H, all_players, winners)
order_likelihood(M,times=1)
```

Arguments

H	Object of class hyper2
...	Numeric or character vectors specifying groups of players with equal rank, with higher-ranking groups coming earlier in the argument list
all_players,winners,losers	Numeric or character vectors specifying (all,winning,losing) competitors in randomly chosen groups
M	In function order_likelihood(), a matrix with each row corresponding to a race. The columns correspond to the finishing order; thus a=M[i,j] means that competitor a finished in place j in race i
times	Vector specifiying the number of times each row is observed

Details

These functions are designed to return likelihood functions, in the form of lists of hyper2() objects, for typical order statistics such as the results of rowing heats or MasterChef tournaments.

Function ggol() is an easily-typed alias for general_grouped_order_likelihood().

Function addrank() is now defunct.

Author(s)

Robin K. S. Hankin

See Also

rrank

Examples

```
W <- hyper2(pnames=letters[1:7])
W1 <- ggol(W, 'a', letters[2:6],'g')  # 24-element list
W2 <- ggol(W, 'b', letters[3:7],'a')  # 24-element list

like_single_list(rep(1/7,6),W1)
like_series(rep(1/7,6),list(W1,W2))

# Run 10 races:
r1 <- rrank(10,p=(7:1)/28)
colnames(r1) <- letters[1:7]

# Likelihood function for r1:
W <- order_likelihood(r1)

# Incorporate information from another (independent) observation:
W <- W + order_likelihood(sample(7))


# consistency check:
H <- hyper2(pnames=letters[1:5])
H <- H + order_likelihood(rrank(100,5:1)) # probs = 5/15,4/15,...,1/15
maxp(H)  # should be close to (5:1)/15
```

`gradient` *Differential calculus*

Description

The gradient of the log-likelihood of a hyper2 object, at a specific point in probability space

Usage

```
gradient(H, probs)
```

Arguments

`H`	A `hyper2` object
`probs`	A vector of probabilities

Details

Function `gradient()` returns the gradient of the log-likelihood function. If the `hyper2` object is of size n, then argument `probs` must be a vector of length $n - 1$.

Note

This functionality is peculiarly susceptible to off-by-one errors.

Author(s)

Robin K. S. Hankin

See Also

differentiate

Examples

```
data(chess)
p <- c(1/2,1/3)
delta <- rnorm(2)/1e5  # delta needs to be quite small

deltaL  <- loglik(chess,p+delta) - loglik(chess,p)
deltaLn <- sum(delta*gradient(chess,p + delta/2))   # numeric

deltaL - deltaLn  # should be small
```

---

handover *Dataset on communication breakdown in handover between physicians*

Description

Object handover is a likelihood function corresponding to a dataset arising from 69 medical malpractice claims and concerns handover (or hand-off) between physicians. This dataset was analysed by Lin et al. (2009), and further analysed by Altham and Hankin (2010). The computational methods are presented in the 'hyperdirichlet' and 'aylmer' packages and a further discussion is given in the "integration" vignette of the 'hyper2' package.

Usage

```
data(handover)
```

Details

Details are provided in the integration vignette, and the cited papers.

References

- Y. Lin and S. Lipsitz and D. Sinha and A. A. Gawande and S. E. Regenbogen and C. C. Greenberg, 2009. "Using Bayesian p-values in a 2×2 table of matched pairs with incompletely classified data". *Journal of the Royal Statistical Society, Series C*, 58:2
- P. M. E. Altham and R. K. S. Hankin, 2010. "Using recently developed software on a 2×2 table of matched pairs with incompletely classified data". *Journal of the Royal Statistical Society, series C*, 59(2): 377-379
- R. K. S. Hankin 2010. "A generalization of the Dirichlet distribution". *Journal of Statistical software*, 33:11
- L. J. West and R. K. S. Hankin 2008. "Exact tests for two-way contingency tables with structural zeros". *Journal of Statistical software*, 28:11

Examples

```
data(handover)
maxp(handover)
```

`head.hyper2` *First few terms of a distribution*

Description

First few terms in a hyperdirichlet distribution

Usage

```
## S3 method for class 'hyper2'
head(x, ...)
```

Arguments

`x`	Object of class `hyper2`
`...`	Further argments, passed to `head()`

Details

Function is `x[head(brackets(x), ...)]`

Value

Returns a `hyper2` object

Author(s)

Robin K. S. Hankin

Examples

```
H <- order_likelihood(rrank(100,p=(10:1)/55))
head(H)
```

---

hyper2 *Basic functions in the hyper2 package*

Description

Basic functions in the hyper2 package

Usage

```
hyper2(L, d, pnames = NA)
## S3 method for class 'hyper2'
brackets(H)
## S3 method for class 'hyper2'
powers(H)
## S3 method for class 'hyper2'
pnames(H)
size(H)
as.hyper2(L,d,pnames)
is_valid_hyper2(L,d,pnames)
is_constant(H)
```

Arguments

H	A hyper2 object
L	A list whose elements specify the brackets of a hyper2 object
d	A vector of powers; `hyper2()` recycles *only if* `d` is of length 1
pnames	A character vector specifying the names of p_1 through p_n.

Details

These are the basic functions of the hyper2 package. Function `hyper()` is the low-level creator function; `as.hyper2()` is a bit more user-friendly and attempts to coerce its arguments into a suitable form; for example, a matrix is interpreted as rows of brackets.

Function `is_valid_hyper2()` tests for valid input, returning a Boolean.

Function `size()` returns the (nominal) length n of nonnegative vector $p = (p_1, \ldots, p_n)$ where $p_1 + \cdots + p_n = 1$. Note that function `lhyper2()` takes the vector $p = (p_1, \ldots, p_{n-1})$.

Author(s)

Robin K. S. Hankin

See Also

Ops.hyper2,Extract,loglik

Examples

```
hyper2(list(1,2,3,1:2,2:3),1:5)
```

---

icons *Dataset on climate change due to O'Neill*

Description

Object `icons_matrix` is a matrix of nine rows and six columns, one column for each of six icons relevant to climate change. The matrix entries show the number of respondents who indicated which icon they found most concerning. The nine rows show different classes of respondents who were exposed to different subsets (of size four) of the six icons.

The columns correspond to the different stimulus icons used, detailed below.

Object `icons` is the corresponding likelihood function, which can be created with `saffy(icons_matrix)`.

Usage

```
data(oneill)
```

Details

The six icons were used in this study were:

PB polar bears, which face extinction through loss of ice floe hunting grounds

NB The Norfolk Broads, which flood due to intense rainfall events

LF London flooding, as a result of sea level rise

THC The Thermo-haline circulation, which may slow or stop as a result of anthropogenic modification of the hydrological cycle

OA Oceanic acidification as a result of anthropogenic emissions of carbon dioxide

WAIS The West Antarctic Ice Sheet, which is calving into the sea as a result of climate change

Source

Data kindly supplied by Saffron O'Neill of the University of East Anglia

References

- S. O'Neill 2007. *An Iconic Approach to Communicating Climate Change*, University of East Anglia, School of Environmental Science (in prep)
- I. Lorenzoni and N. Pidgeon 2005. *Defining Dangers of Climate Change and Individual Behaviour: Closing the Gap*. In *Avoiding Dangerous Climate Change* (conference proceedings), UK Met Office, Exeter, 1-3 February
- R. K. S. Hankin 20101. "A generalization of the Dirichlet distribution". *Journal of Statistical software*, 33:11

Examples

```
data(oneill)
maxp(icons)

icons == saffy(icons_matrix)   # should be TRUE
```

`increment` *Increment and decrement operators*

Description

Syntactic sugar for incrementing and decrementing likelihood functions

Usage

```
inc(H, val = 1)
dec(H, val = 1)
trial(H,winners,players,val=1)
```

Arguments

`H`	A hyper2 object
`winners,players`	Numeric or character vectors specifying the winning team and the losing team
`val`	Numeric

Details

A very frequent operation is to increment a single term in a hyper2 object. If

```
R> H <- hyper2(list(1,2,3,1:2,2:3),1:5)
R> pnames(H) <- letters[1:3]
```

Then we would have

```
R> H
a * (a + b)^4 * b^2 * (b + c)^5 * c^3
```

Suppose we wish to increment the power of `a+b`. We could do:

```
H[c("a","b")] <- H[c("a","b")] + 1
```

Or use `magrittr` pipes:

```
H[c("a","b")] %<>% `+`(1)
```

But inc and dec furnish convenient idiom to accomplish the same thing:

```
H[c("a","b")] %<>% inc
```

Functions inc and dec default to adding or subtracting 1, but other values can be supplied:

```
H[c("a","b")] %<>% inc(3)
```

Or even

```
H[c("a","b")] %<>% inc(H["a"])
```

The convenience function trial() takes this one step further and increments the 'winning team' and decrements the 'losing team'. This function is intended to be used with magrittr pipes:

```
> H <- hyper2(pnames=letters[1:6])
> H
(a + b + c + d + e + f)^0
> H %<>% trial(c("a","b"),c("a","b","c"))
> H
(a + b) * (a + b + c)^-1
```

Using trial() in this way ensures that the powers sum to zero.

The inc and dec operators are used in inst/rowing_analysis.R; and the trial() function is used in inst/kka_draws.R.

Author(s)

Robin K. S. Hankin

Examples

```
data(chess)

## Now suppose Topalov beats Anand. To  incoroprate this into the LF:
chess["Topalov"] %<>% inc
chess[c("Topalov","Anand")] %<>% dec
chess
```

interzonal *1963 World Chess Championships*

Description

Likelihood functions for players' strengths in the fith Interzonal tournament which occurred as part of the 1963 Chess world Championships in Stockholm, 1962.

Details

The 1963 World Chess Championships was notable for allegations of Soviet collusion. Specifically, Fischer publicly alledged that certain Soviet players had agreed in advance to draw all their games.

Likelihood functions `interzonal` and `interzonal_collusion` are created by files `inst/collusion_stockholm.R`, which are heavily documented and include some analysis. Object `interzonal` includes a term for drawing, ("`draw`"), assumed to be the same for all players; object `collusion` includes in addition to `draw`, a term for the drawing in Soviet-Soviet matches, "`coll`".

See Also

`chess`,`karpov_kasparov_anand`

Examples

```
data(interzonal)
dotchart(maxp(interzonal))

maxp(collusion)
```

karpov_kasparov_anand *Karpov, Kasparov, Anand*

Description

Data of three chess players: Karpov, Kasparov, and Anand. Includes two likelihood functions for the strengths of the players, and matrices of game results

Details

The strengths of chess players may be assessed using the generalized Bradley-Terry model. The `karpov_kasparov_anand` likelihood function allows one to estimate the players' strengths, propensity to draw, and also the additional strength conferred by playing white.

Likelihood functions `karpov_kasparov_karpov`, `kka_3draws` and `kka_3whites` are created by files `inst/karpov_kasparov_anand.R`, `inst/kka_3draws` and `inst/kka_3whites`, which are heavily documented and include some analysis. Object `karpov_kasparov_anand` assumes that the draw

potential is the same for all three players; likelihood function kka_3draws allows the propensity to draw to differ between the three players.

The reason that the players are different from those in the chess dataset is that the original data does not seem to be available any more.

Dataset kka refers to scorelines of matches between three chess players (Kasparov, Karpov, Anand). It is a list with names such as 'karpov_plays_white_beats_kasparov' which has value 18: we have a total of 18 games between Karpov and Kasparov in which Karpov played white and beat Kasparov.

The three matrices plays_white_wins, plays_white_draws, and plays_white_loses tabulate this information in a coherent way; and array kka_array presents the same information in a 3D array (but the names of the dimnames are lost).

All data drawn from chessgames.com, specifically

http://www.chessgames.com/perl/ezsearch.pl?search=karpov+vs+kasparov

Note that the database allows one to sort by white wins or black wins (there is a 'refine search' tab at the bottom). Some searches have more than one page of results.

Numbers here downloaded 17 February 2019. Note that only 'classical games' are considered here (rapid and exhibition games being ignored).

See Also

chess

Examples

```
data(karpov_kasparov_anand)
maxp(karpov_kasparov_anand)
pie(maxp(karpov_kasparov_anand))
```

---

length.hyper2 *Length method for hyper2 objects*

Description

Length method for hyper2 objects, being the number of different brackets in the expression

Usage

```
## S3 method for class 'hyper2'
length(x)
```

Arguments

x	hyper2 object

Author(s)

Robin K. S. Hankin

Examples

```
data("oneill")
length(icons)
seq_along(icons)
```

`loglik` *Log likelihood functions*

Description

Returns a log-likelihood for a given hyper2 object at a specific point

Usage

```
loglik(H, p, log = TRUE)
like_single_list(p,Lsub)
like_series(p,L,log=TRUE)
```

Arguments

`H`	An object of class `hyper2`
`p`	A probability point. See details
`log`	Boolean with default `TRUE` meaning to return the log-likelihood and `FALSE` meaning to return the likelihood
`L,Lsub`	A list of `hyper2` objects, or a list of list of `loglik` objects

Details

Function `loglik()` is a straightforward likelihood function. It takes the vector $p = (p_1, \ldots, p_{n-1})$ and returns the (log) likelihood. Note that the 'fillup' value is automatically appended; the elements of `p` are (linearly!) independent.

Function `size()` returns the (nominal) length n of nonnegative vector $p = (p_1, \ldots, p_n)$ where $p_1 + \cdots + p_n = 1$.

If `p` is a matrix, the rows are interpreted as probability points.

Functions `like_single_list()` and `like_series()` are intended for use with `ggol()`.

Note

Likelihood is defined up to an arbitrary multiplicative constant.

Log-likelihood (also known as *support*) is defined up to an arbitrary additive constant.

Author(s)

Robin K. S. Hankin

See Also

maxp

Examples

```
data(chess)
loglik(chess,c(1/3,1/3))

data(masterchef)

a1 <- rep(1/13,12)                          # equal strengths
a2 <- indep(pmax_masterchef6)               # MLE

like_series(a1,masterchef_series6)
like_series(a2,masterchef_series6)


W <- hyper2(pnames=letters[1:6])
W1 <- ggol(W, 'a', letters[2:5],'f')              # 24-element list
W2 <- ggol(W, c('a','b'), c('c','d'),c('e','f'))  # 2^3=8 element list

like_single_list(rep(1/6,5),W1)      # information from first observation
like_series(rep(1/6,5),list(W1,W2))  # information from both observations
```

---

masterchef *Masterchef series 6*

Description

Data from Australian Masterchef Series 6

Usage

```
data(masterchef)
```

Format

Object masterchef_series6 is a list of hyper2 objects; pmax_masterchef_series6 and pmax_masterchef_series6_cons
are named vectors with unit sum.

Details

The object is created using the code in `inst/masterchef_series6_allrounds.R`, which is heavily documented. Not all the information available is included in the likelihood function as some of the early rounds result in an unmanageably large list. Inclusion is controlled by Boolean vector `doo`.

The definitive source is the coloured table on the wiki page.

References

Wikipedia contributors, "MasterChef Australia (series 6)," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=MasterChef_Australia_(series_6)&oldid=758432561 (accessed January 5, 2017).

See Also

`ggol`

Examples

```
data(masterchef)

a1 <- rep(1/13,12)                          # equal strengths
a2 <- indep(pmax_masterchef6)               # MLE
a3 <- indep(pmax_masterchef6_constrained)   # constrained MLE

like_series(a1, masterchef_series6)
like_series(a2, masterchef_series6)
like_series(a3, masterchef_series6)
```

---

maxp *Maximum likelihood estimation*

Description

Find the maximum likelihood estimate for p, also equal probabilities

Usage

```
maxp(H, startp=NULL, give=FALSE, fcm=NULL, fcv=NULL, ...)
equalp(H)
```

Arguments

`H`	A `hyper2` object
`startp`	A vector of probabilities
`give`	Boolean, with default `FALSE` meaning to return just the evaluate (including fillup), and `TRUE` meaning to return the entire formal output of the optimization routine
`fcm,fcv`	Further problem-specific constraints
`...`	Further arguments which `maxp()` passes to `constrOptim()`

Details

Function `maxp()` returns the maximum likelihood estimate for `p`, which has the unit sum constraint implemented. The function does not work for the output of `ggol()` nor the `masterchef_series6` dataset. These require a bespoke optimization as shown in the vignette.

Function `equalp()` returns the value of p for which all elements are the same.

In function `maxp()`, arguments `fcm` and `fcv` implement linear constraints to be passed to `constrOptim()`. These constraints are in addition to the usual nonnegativity constraints and unit-sum constraint. Examples of their use are given in the "icons" vignette.

Note

This functionality is peculiarly susceptible to off-by-one errors.

Author(s)

Robin K. S. Hankin

See Also

gradient,fillup

Examples

```
data(chess)
maxp(chess)


data(rowing)

(x1 <- loglik(sculls2016,indep(maxp  (sculls2016))))
(x2 <- loglik(sculls2016,indep(equalp(sculls2016))))



## get a p-value for the null of equal player strengths:
pchisq(2*(x1-x2),df=size(sculls2016)-1,lower.tail=FALSE)

## Note that Wilks's theorem is only asymptotic.
```

---

mult_grid *Kronecker matrix functionality*

Description

Peculiar version of `expand.grid()` for matrices

Usage

```
mult_grid(L)
pair_grid(a,b)
```

Arguments

L	List of matrices
a,b	Matrices

Details

Function pair_grid(a,b) returns a matrix with each column of a cbind()-ed to each colum of b.

Function mult_grid() takes a list of matrices; it is designed for use by ggol().

Author(s)

Robin K. S. Hankin

See Also

ggol

Examples

```
pair_grid(diag(2),diag(3))
mult_grid(lapply(1:4,diag))
```

NBA *Basketball dataset*

Description

A point-by-point analysis of a basketball game

Usage

```
data(NBA)
```

Details

Dataset NBA is a dataframe contains a point-by-point analysis of a basketball match. Each row corresponds to a point scored. The first column is the time of the score, the second is the number of points scored, the third shows which team had possession at the start of play, and the fourth shows which team scored.

The other columns show the players. Table entries show whether or not that particular player was on the pitch when the point was scored.

Likelihood function NBA_likelihood is a hyper2 object that gives the log-likelihood function for this dataset. It may be created by executing basketball.R, in the inst/ directory.

Note that the two "ghost" players represent the effect of having possession.

References

- http://www.espn.com/nba/playbyplay?gameId=400954514
- https://www.nbafullhd.com/24828-2/
- http://www.NBAbase.com/

Examples

```
data(NBA)
dotchart(maxp(NBA_likelihood))
```

Ops.hyper2 *Arithmetic Ops Group Methods for hyper2 objects*

Description

Allows arithmetic operators "+", "*" and comparison operators "==" and "!=", to be used for hyper2 objects.

Specifically, H1 + H2 implements addition of two log-likelihood functions, corresponding to incorporation of new observational data; and n*H1 implements H1+H1+...+H1, corresponding to repeated observations of the same data.

There are no unary operations for this class.

Usage

```
## S3 method for class 'hyper2'
Ops(e1, e2 = NULL)
## S3 method for class 'hyper2'
sum(x,...,na.rm=FALSE)
hyper2_add(e1,e2)
hyper2_equality(e1,e2)
hyper2_sum_numeric(H,r)
```

Arguments

e1,e2	Objects of class hyper2, here interpreted as hyperdirichlet distributions
x,...,na.rm	In the sum() method, objects to be summed; na.rm is currently ignored
H,r	In function hyper2_sum_numeric(), object H is a hyper2 object and r is a length-one real vector (a number)

Details

Testing for equality is not straightforward for two implementation reasons. Firstly, the object itself is stored internally as a `stl map`, which does not store keys in any particular order; and secondly, the `stl set` class is used for the brackets. A set does not include information about the order of its elements; neither does it admit repeated elements. See examples.

Function `hyper2_sum_numeric()` is defined so that idiom like

```
H[p] <- H[p] + 3
```

works (without this, one has to type `H[p] <- powers(H[p]) + 3`, which sucks).

Value

Returns a `hyper2` object or a Boolean.

Author(s)

Robin K. S. Hankin

Examples

```
stopifnot(hyper2(list(1,1:2),1:2)==hyper2(list(1:2,1),2:1))  # stl map class
stopifnot(hyper2(list(1,1:2),1:2)==hyper2(list(1,c(1,1,2)),1:2)) # stl set class


n <- 4
H <- hyper2(d=n)
for(i in seq_len(30)){
  jj <- sample(seq_len(n),sample(n,1),replace=FALSE)
  H[jj] <- H[jj] + 1
}


## without pnames:
x1 <- hyper2(list(1,1:2,1:3),1:3)
x2 <- hyper2(list(4,5),10:11)
x1+x2

## Same as above, but with pnames:
y1 <- hyper2(list(1,1:2,1:3),1:3,pnames=letters[1:3])
y2 <- hyper2(list(4,5),10:11,pnames=letters[1:5])
## Not run: y1+y2  # fails, pnames not identical

pnames(y1) <- letters[1:5]
y1+y2
```

Print *Print methods*

Description

Print methods for hyper2 objects

Usage

```
## S3 method for class 'hyper2'
print(x, ...)
```

Arguments

x	An object of class hyper2
...	Further arguments, currently ignored

Value

Returns the hyper2 object it was sent, invisibly. Used mainly for its side-effect

Note

This is the only place that the pnames attribute is used

Author(s)

Robin K. S. Hankin

Examples

```
data(chess)
chess
```

rowing *Rowing dataset, sculling*

Description

Data from Men's single sculls, 2016 Summer Olympics

Usage

```
data(rowing)
```

Format

sculls2016 is a hyper2 object that gives a likelihood function for the 2016 men's sculls.

Details

Object sculls2016 is created by the code in inst/rowing_analysis.R. This reads file inst/rowing.txt, each line of which is a heat showing the finishing order. Character vector allrowers holds the names of all the rowers.

File inst/rowing_minimal.txt has the same data but with dominated players (that is, any group of players none of whom have beaten any player not in the group) have been removed. This is because dominated players have a ML strength of zero.

References

Wikipedia contributors, "Rowing at the 2016 Summer Olympics—Men's single sculls", *Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Rowing_at_the_2016_Summer_Olympics_%E2%80%93_Men%27s_single_sculls&oldid=753517240 (accessed December 7, 2016).

See Also

ggol

Examples

```
data(rowing)
dotchart(maxp(sculls2016))
```

---

rrank *Random ranks*

Description

A function for producing ranks randomly, consistent with a specified strength vector

Usage

```
rrank(n = 1, p, pnames=NULL, fill = FALSE)
## S3 method for class 'rrank'
print(x, ...)
```

Arguments

n	Number of observations
p	Strength vector
pnames	Character vector specifying names of the columns
fill	Boolean, with default FALSE meaning to interpret the elements of p as strengths, notionally summing to one; and TRUE meaning to augment p with a fillup value
x, ...	Arguments passed to the print method

Value

If n=1, return a vector; if n>1 return a matrix with n rows, each corresponding to a ranking. The canonical example is a race in which the probability of competitor i coming first is $p_i / \sum p_j$, where the summation is over the competitors who have not already finished.

If, say, the first row of rrank() is c(2,5,1,3,4), then competitor 2 came first, competitor 5 came second, and so on.

Note that function rrank() returns an object of class rrank, which has its own special print method. The column names appear as "c1, c2, ..." which is intended to be read "came first", "came second", and so on. The difference between *rank* and *order* can be confusing.

```
R> x <- c( 3, 1, 2, 4.1)
R> rank(x)
[1] 3 1 2 4
R> order(x)
[1] 2 3 1 4
R>
```

In the current context, we have:

```
R> rrank(2,ptrue)
     c1 c2 c3 c4
[1,]  3  1  2  4
[2,]  2  3  4  1
R>
```

In the above, each row is a race; we have four runners. In the first race, runner number 3 came first, runner 1 came second, runner 2 came third, and so on. In the second race, runner 2 came first, etc. Now consider the following R idiom which uses the first race:

```
R> p <- c(3,1,2,4)
R> order(p)    # slicker to say p[p] <- 1:4 as in inverse_word_single()
[1] 2 3 1 4
R>
```

This shows that runner 1 came second, runner 2 came third, runner 3 came first, and runner 4 came fourth.

Take the first race as an example.

Rank: who came first? runner 3. Who came second? runner 1. Who came third? runner 2. Who came fourth? runner 4.

Order: where did runner 1 come? second. Where did runner 2 come? third. Where did runner 3 come? first. Where did runner 4 come? fourth. For `order_likelihood()`, we need rank data, not order data. See vignette "`skating_analysis`" for more discussion.

Author(s)

Robin K. S. Hankin

See Also

`ggol`,`order_likelihood`,`skating`

Examples

```
ptrue <- (4:1)/10
rrank(100,p=ptrue)

r1 <- rrank(100,p=ptrue)
r2 <- rrank(10,p=ptrue)

H <- order_likelihood(r1)
maxp(H)   # should be close to ptrue

H <- H + order_likelihood(rrank(100,maxp(H)))
maxp(H)
```

---

rugby *Super Rugby 2016 data*

Description

Data from 2016 Super Rugby season. Two `hyper2` objects, `super2016` and `super2016g`. The second includes a "ghost" team whose strength helps the home team.

Usage

```
data(super_rugby)
```

References

- Wikipedia contributors, "Super Rugby," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Super_Rugby&oldid=758848475 (accessed January 7, 2017).
- Wikipedia contributors, "2016 Super Rugby season," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=2016_Super_Rugby_season&oldid=758243138 (accessed January 4, 2017).

See Also

loglik

Examples

```
data(super_rugby)
loglik(super2016,indep(maxp(super2016)))
loglik(super2016g,indep(maxp(super2016g)))

# differ by more than 2, ghost is signficant!
```

---

saffy *Choice matrix functionality*

Description

Converts a matrix of restricted choices into a likelihood function. It is named for Saffron O'Neill.

Usage

```
saffy(M)
```

Arguments

M	Matrix

Details

The canonical example is Saffron's climate change dataset, documented at `icons`. Function `saffy()` returns the appropriate likelihood function for the dataset.

Author(s)

Robin K. S. Hankin

See Also

icons

Examples

```
data("oneill")
icons == saffy(icons_matrix)  # should be TRUE
```

---

skating *Figure skating at the 2002 Winter Olympics*

Description

A likelihood function for the competitors at the Ladies' Free Skate at the 2002 Winter Olympics

Usage

```
data(skating)
```

Details

Object `skating` can be generated by running script `inst/skating_analysis.R`, which includes some further technical documentation.

The dataset is interesting because it has been analysed by many workers, including Lock and Lock, for consistency between the judges.

Note that file is structured so that each competitor is a row, and each judge is a column. Function `order_likelihood()` requires a transpose of this to operate.

References

- https://en.wikipedia.org/wiki/Figure_skating_at_the_2002_Winter_Olympics#Full_results_2
- Robin Lock and Kari Frazer Lock, Winter 2003. "Judging Figure Skating Judges". *STATS 36*, ASA

Examples

```
data(skating)
dotchart(maxp(skating))

par(pty='s')
lik <- order(maxp(skating),decreasing=TRUE)
plot(1:23,lik,asp=1,pch=16,xlab='official order',ylab='likelihood order')
text(1:23,lik,pos=c(rep(4,19),rep(2,4)),pnames(skating))
abline(0,1)
```

table_tennis *Match outcomes from repeated table tennis matches*

Description

Match outcomes from repeated singles table tennis matches

Usage

```
data(table_tennis)
```

Format

A likelihood function corresponding to the match outcomes listed below.

Details

There are four players, A, B, and C, who play singles table tennis matches with the following results:

- A vs B, A serves, 5-1
- A vs B, B serves, 1-3
- A vs C, A serves, 4-1
- A vs C, C serves, 1-2

As discussed in vignette `table_tennis_serve`, we wish to assess the importance of the serve. The vignette presents a number of analyses including a profile likelihood plot.

See vignette `table_tennis_serve` for an account of how to create `table_tennis`.

Examples

```
data(table_tennis)
dotchart(maxp(table_tennis))
```

tennis *Match outcomes from repeated doubles tennis matches*

Description

Match outcomes from repeated doubles tennis matches

Usage

```
data(tennis)
```

Format

A hyper2 object corresponding to the match outcomes listed below.

Details

There are four players, p_1 to p_4. These players play doubles tennis matches with the following results:

match	score
$\{p_1, p_2\}$ vs $\{p_3, p_4\}$	9-2
$\{p_1, p_3\}$ vs $\{p_2, p_4\}$	4-4
$\{p_1, p_4\}$ vs $\{p_2, p_3\}$	6-7
$\{p_1\}$ vs $\{p_3\}$	10-14
$\{p_2\}$ vs $\{p_3\}$	12-14
$\{p_1\}$ vs $\{p_4\}$	10-14
$\{p_2\}$ vs $\{p_4\}$	11-10
$\{p_3\}$ vs $\{p_4\}$	13-13

It is suspected that p_1 and p_2 have some form of team cohesion and play better when paired than when either solo or with other players. As the scores show, each player and, apart from p1-p2, each doubles partnership, is of approximately the same strength.

Dataset `tennis` gives the appropriate likelihood function for the players' strengths; and dataset `tennis_ghost` gives the appropriate likelihood function if the extra strength due to team cohesion of $\{p_1, p_2\}$ is represented by a ghost player.

Source

Doubles tennis matches at NOCS, Jan-May 2008

References

Robin K. S. Hankin (2010). "A Generalization of the Dirichlet Distribution", *Journal of Statistical Software*, 33(11), 1-18, http://www.jstatsoft.org/v33/i11/

Examples

```
data(tennis)
dotchart(maxp(tennis))
eigen(maxp(tennis,give=TRUE,hessian=TRUE)$hessian,TRUE,TRUE)$values

maxp(tennis_ghost)
```

`tidy` *Keep or discard players*

Description

Functionality to keep or discard subsets of the players in a hyper2 object

Usage

```
tidy(H)
keep(H, keep, tidy=TRUE)
discard(H, discard, tidy=TRUE)
```

Arguments

`H`	A `hyper2` object
`tidy`	Boolean, with default `TRUE` meaning to return a mathematically identical, but tidied, likelihood function
`keep,discard`	Players to keep or discard. May be character or integer

Details

Function `tidy(H)` returns a hyper2 object mathematically identical to `H` but with unused players (that is, players that do not appear in any bracket) removed.

Functions `keep()` and `discard()` will either keep or discard players specified in the second argument.

Note

Function `tidy()` is very, very, inelegant

Author(s)

Robin K. S. Hankin

Examples

```
data("oneill")
maxp(icons)
discard(icons,c("OA","WAIS"))

data("skating")
maxp(skating)[1:4]      # numbers work, keep the first four skaters
maxp(keep(skating,1:4))
```

`volley` *Team sports*

Description

Converts a matrix of games into a likelihood function.

Usage

```
volley(M)
```

Arguments

`M`	Matrix

Details

The canonical example is the volleyball dataset. Each row is a volleyball game; each column is a player. An entry of 0 means "on the losing side", an entry of 1 means "on the winning side", and an entry of `NA` means `did not play`.

Author(s)

Robin K. S. Hankin

See Also

`saffy`

Examples

```
data(volleyball)
jj <- volley(volleyball_matrix)
stopifnot(jj == volleyball)
```

Index